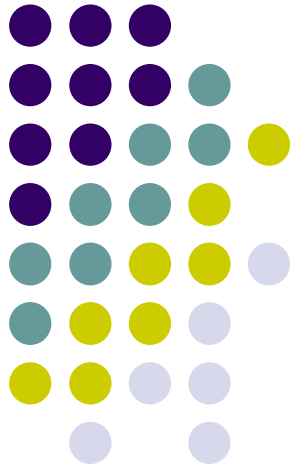
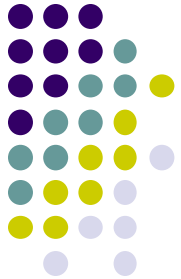
WIX1002
Fundamentals of Programming

Chapter 10

Polymorphism

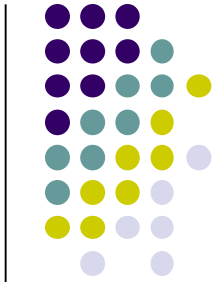




Contents

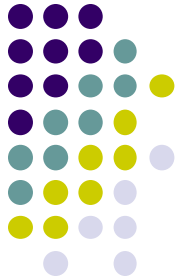
- Polymorphism
- clone Method
- Abstract Classes
- Interface
- Comparable Interface
- Inner Class

Polymorphism



- **Polymorphism** refers to the ability to associate many meanings to one method name using late binding or dynamic binding
- **Binding** refers to the process of associating a method definition with a method invocation.
- **Early Binding** associate a method definition with the method invocation when the code is compiled.
- **Late binding / Dynamic Binding** associate a method definition with the method invocation when the method is invoked (**at run time**)
- Java does not use late binding with private method, methods with final modifier and static method.

Polymorphism



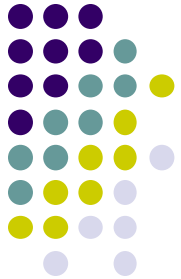
- An object of a derived class has the type of its base class.
- Assigning an object of a derived class to a variable of base classis often called **upcasting**.

  BaseClass a = new BaseClass;

  DerivedClass b = new DerivedClass;

  a = b;

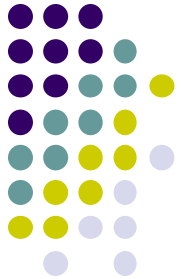
Polymorphism



- Downcasting assigning an object of a base class to a derived class.
- To test whether the downcasting is legitimate

```
DerivedClass obj = new DerivedClass();
if (obj1 instanceof DerivedClass) {
   obj = (DerivedClass) obj1
```

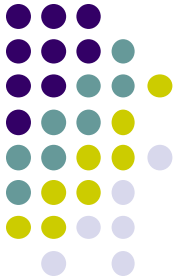
clone Method



- Every object inherits a method named clone from the class Object.
- The clone method **return a copy of the calling object**.

```
public ClassName clone() {
   return new ClassName(this);
}

public Sale doCopy(Sale a) {
   Sale b;
   b = a.clone();
}
```

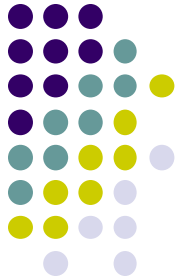
Abstract Classes



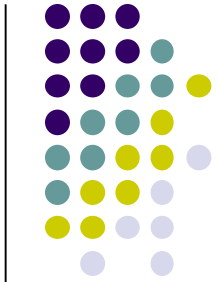
- An abstract class is a class that has **some methods without complete definitions**. A class with no abstract methods is called a concrete class.
  - public abstract class className
- Object can’t be created using abstract class constructor.
- An **abstract method** serves as a placeholder for a method that will be fully defined in a descendent class. It has **no method body**.
- An abstract method can’t be private
  - public abstract returnType methodName(parameterType parameterName, …);

Abstract Classes

```
public abstract class Employee {
   public abstract double getPay();
   ...
}

Employee a = new Employee()
// can't create object for an abstract class
```



Interface



- A Java interface specifies a set of methods that any class that implements the interface must have.
- An **interface** is a type that groups together a number of different classes that all include method definitions for a common set of method headings.
  - public interface interfaceName
- An interface has **no instance variable** and **no method definitions**.
- An interface can contain defined **constants** as well as **method headings**. When a method implements the interface, it automatically gets the defined constant.

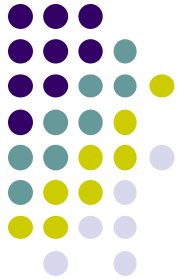
Interface

```
public interface interfaceName {

    public static final int MAX = 100;

    public returnType methodName(parameterType
     parameterName, ...);

}
```

Interface

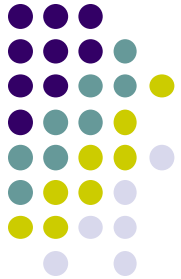
- The class must **implement all the method headings** listed in the definitions of the interfaces.
  - public class className **implements interfaceName, …**
- When a class implements an interface, it must make all the methods in the interface public.
- In Java, a derived class can have only **one base class**. However, a class can implement any number of interfaces.

Comparable Interface



- Comparable interface has one method
  - **public int compareTo(Object other);**
  - The method return negative if calling object comes before the parameter object
  - The method return positive if calling object comes after the parameter object
  - The method return zero if calling object equals to the parameter object

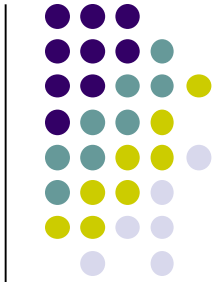
Inner Class



- Inner class are classes defined within other classes.
- Inner and outer classes have access to each other's private members

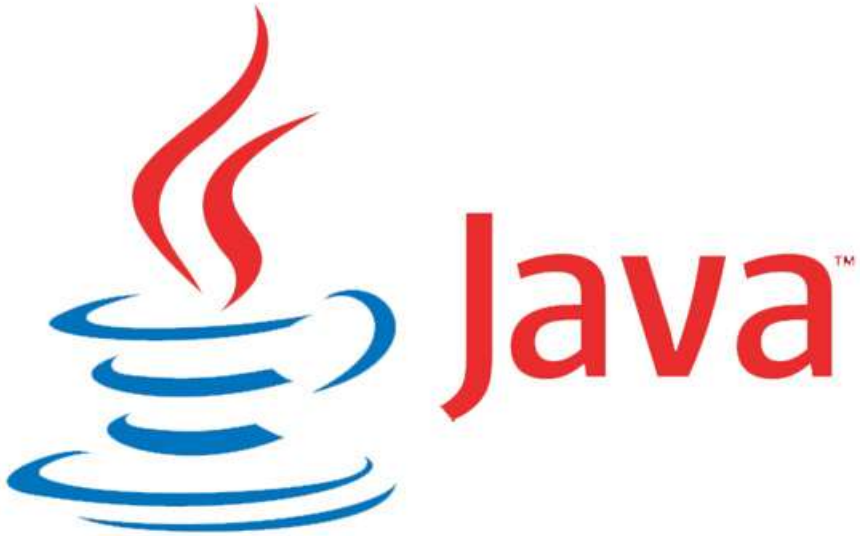
```
public class ClassName {

   private class ClassName {

   }

}
```

Inner Class



- A static Inner class **can have nonstatic instance variables and methods** but an object of static inner class has no connection to an object of the outer class.

```
public class ClassName {

   private static class ClassName {

   }

}
```


Java™